A very important data structure is the *stack*. A stack works according to the LIFO principle, i.e., last in first out:

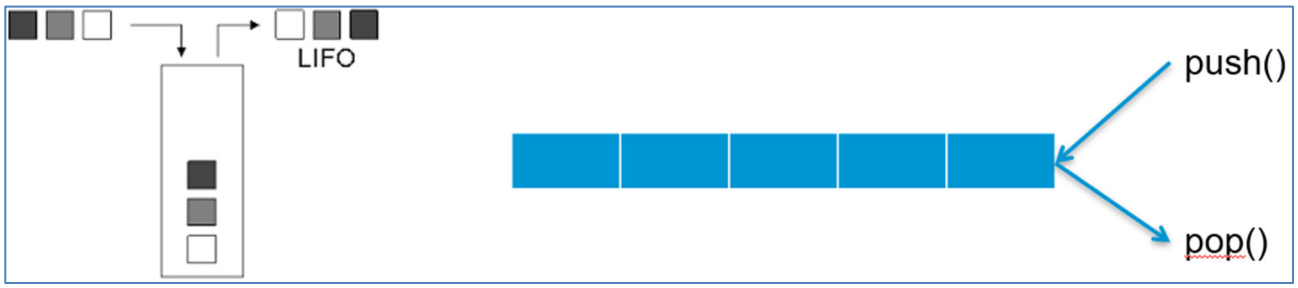

Figure 1: How a stack works

(Source: http://oa.upm.es/47183/1/TFG_MARIO_MARTIN_RICOTE.pdf)

This data structure is used in many ways. For instance, when you run a C program on your computer, all the data the program needs are stored in the RAM. The execution stack in the RAM is essential for storing all variables, return addresses to functions, etc..

In this assignment, we will focus on a simpler example, namely a car transporter. A car transporter works as follows: The car that is loaded first on the transporter is unloaded last. And the car that is loaded on the transporter last is the first to be unloaded.

Figure 2: Car transporter

(Source: https://www.ebay-kleinanzeigen.de/s-autotransporter/k0)

For each car on the car transporter, the brand of the car and the license plate number should be stored as an example. The data type for a car in C could look as follows.

```
typedef struct sCar {
    char brand[MAXSTRING];
    char licensePlate[MAXSTRING];
} sCar;
```

Figure 3: Data Type sCar

The car transporter is now to be implemented by means of a stack. In the following assignments the data structure stack is to be implemented, first using an array, and then using a singly linked list. The following typical functions for the data structure stack must be programmed:

- `push()`: Puts a new element on the stack.
  In our example a new car is loaded onto the car transporter.
- `pop()`: Removes the last element from the stack.
  In our example the last loaded car will be unloaded.
- `isEmpty()`: Checks if the stack has at least one element.
  In our example, the function checks if there is at least one car on the car transporter.
- `countElements()`: Counts the elements in the stack.
  In our example, the function counts how many cars are on the car transporter.
- `printStack()`: Prints all the elements on the console that are stored in the stack.
  In our example it prints all brands and license plates of the loaded cars.

In our moodle room you can find templates for both the stack and the singly linked list implementation.

Assignment 1: Array Implementation

In the first assignment, the stack is to be implemented with an array. The array should be able to store 50 cars. Since the number of cars should be flexibly adjustable, define the macro `MAXARRAY` for the number. In the laboratory, we will do the programming for the functions `push()`, `countElements()`, and `printStack()` together.

Assignment 1.1: Implementing function `pop()`

Implement the `pop()` function that prints the top element to the console and then removes it from the stack.
The function does not have a return value.

Note: Think also about special cases, such as that the stack is empty.

Assignment 1.2: Implementing function `isEmpty()`

Implement the function `isEmpty()` that checks if the car transporter is empty:

- If the car transporter is empty, the function should return 0.
- If there are cars on the car transporter, the function should return 1.

Assignment 2: Singly linked list Implementation

Now we implement the stack with a singly linked list. In the laboratory, we will do the programming for the functions `countElements()` and `printStack()` together.

Assignment 2.1: Implementing function `push()`

Implement the `push()` function that first creates a new car. To do this, the function asks the user for the car brand and license plate. Second, it adds this car to the stack and returns this element as the new top element of the stack.

Note: Think also about special cases, such as that the stack is empty.

Assignment 2.2: Implementing function `pop()`

Implement the `pop()` function that prints the top element to the console and then removes it from the stack.
The function does not have a return value.

Note: Think also about special cases, such as that the stack is empty.

Assignment 2.3: Implementing function `isEmpty()`

Implement the function `isEmpty()` that checks if the car transporter is empty:

- If the car transporter is empty, the function should return 0.
- If there are cars on the car transporter, the function should return 1.